2006 CCRTS
The State of the Art and the State of the Practice

1. Title: **On Applying Point-Interval Logic to Criminal Forensics**[•]

2. Topic Area: C2 Modeling and Simulation

3. Authors: **Mashhood Ishaque** **Abbas K. Zaidi** **Alexander H. Levis**

4. Point of Contact: **Mashhood Ishaque**

5. Organization: System Architectures Lab, George Mason University

6. Address: System Architectures Laboratory
MSN 4B5
George Mason University
Fairfax, VA - 22030
703.993.1725 (v)
703.993.1706 (f)

7. Email: mishaque@acm.org

Student Paper

[•] The work was carried out with support provided by the Air Force Office of Scientific Research under contract numbers FA9550-05-1-0106.

Report Documentation Page		*Form Approved* *OMB No. 0704-0188*
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.		
1. REPORT DATE **2006**	2. REPORT TYPE	3. DATES COVERED **00-00-2006 to 00-00-2006**
4. TITLE AND SUBTITLE **On Applying Point-Interval Logic to Criminal Forensics**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **George Mason University,System Architectures Laboratory,C3I Center,Fairfax,VA,22030**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES **The original document contains color images.**		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**15**	

On Applying Point-Interval Logic to Criminal Forensics

Mashhood Ishaque

Abbas K. Zaidi

Alexander H. Levis

System Architectures Lab
George Mason University
Fairfax, VA – 22030
mishaque@acm.org, {szaidi2, alevis}@gmu.edu

Abstract

Application of a temporal logic to forensic analysis, especially in answering certain investigative questions relating to time-sensitive information, is presented. A set of temporal facts is taken from the London bombing incident that took place on July 7, 2005, to illustrate the approach. The information used in the illustration is gathered through the online news sites. A hypothetical investigation on the information is carried out to identify certain time intervals of potential interest to crime investigators. A software tool called Temper that implements temporal logic is used.

1 Introduction

The growing need for a formal logic of time for modeling and analyzing temporal information has led to the emergence of various types of representations and reasoning schemes. Point-Interval Logic (PIL) is one such formalism. It has been implemented in the form of a software tool called Temper. The expressive language of PIL, combined with a powerful verification, inference, and revision mechanism, make Temper a useful tool for forensics, especially for analyzing the time-sensitive information involved. It can be used to make sense of, or identify, contradictions and inconsistencies in the temporal information available about a particular incident. The verification mechanism of PIL can be used to check the consistency of the available temporal information. The inference mechanism can be used to query the relationships between various temporal events. The efficient revision mechanism of PIL enables what-if analysis. Thus, investigators can test their theories about how some particular incident might have unfolded on a timeline using Temper. In this paper we demonstrate how Temper can be used for criminal forensics by using the scenario of the London bombing incident of July 7, 2005.

Point-Interval Logic (PIL) is a specialization of Pointisable Algebra (Ladkin and Maddux 1988). It originated from an earlier work on temporal knowledge representation and reasoning by Zaidi (1999). The earlier version of PIL was an extension of Hamblin's time primitives (1972) and Allen's Interval logic (1983). This extension allowed the inclusion of points (i.e., intervals with zero lengths) in Allen's ontology. The formalism presented an axiomatic system for the logic. A Petri net (Peterson, 1981; Reisig, 1991) model was shown to represent this axiomatic system by transforming the system's specifications (i.e. qualitative temporal relations between system entities) given by statements of Point-Interval Logic into Petri net structures. The Petri net structure, with some of its analytical tools was subsequently renamed Point Graph (PG). An inference engine based on this Point Graph representation infers new temporal relations among system intervals, identifies temporal ambiguities and errors (if present) in the system's specifications, and finally identifies the intervals of interest defined by the user. Zaidi and Levis (2001) further extended the point-interval approach by adding provisions for "dates/clock" times and time "distances" for points and intervals. This extension allowed the assignment of actual lengths to intervals, time distances between points, and time stamps to points representing the actual time of occurrences, whenever such information is available. A temporal model may change during and/or after the system specification phase. Support for an on-the-fly revision (add, delete, modify) was added to Point Graph formalism in Rauf and Zaidi (2002). Zaidi and Wagenhals (2006) consolidated the results of the previous work on the logic and its application to the modeling and planning time-sensitive aspects of a mission and extended the approach further. The extension allows for a larger class of temporal systems to be handled by incorporating an enhanced input lexicon, allowing increased flexibility in temporal specifications, providing an improved verification and inference mechanism, and adding a suite of analysis tools.

The paper is organized as follows: In Section 2, we present a brief discussion on Point-Interval Logic (PIL). In Section 3, we briefly describe the user interface of Temper (software implementation of PIL). We illustrate modeling situations arising in forensics in Section 4 using the London bombing incident as the scenario. In Section 5, we comment on the contribution of this paper.

2 Point-Interval Logic

The lexicon of the Point Interval Logic (PIL) consists of the following primitive symbols:

Points: A point X is represented as [pX, pX] or simply [pX].

Intervals: An interval X is represented as [sX, eX], where 'sX' and 'eX' are the two end points of the interval, denoting the 'start' and 'end' of the interval, such that sX < eX.

Point Relations: These are the relations that can exist between two points. The set of relations R_P is given as:

$$R_P = \{<, =, \leq\} \text{ or } R_P = \{\text{less than, equal, less-than-or-equal}\}$$

Interval Relations: These are the atomic relations that can exist between two intervals. The set of relations R_I is given as:

$$R_I = \{<, m, o, s, d, f, =\} \text{ or}$$

$$R_I = \{\text{less than, meet, overlap, start, during, finish, equal}\}$$

Point-Interval Relations: These are the atomic relations that can exist between a point and an interval. The set of relations R_{PI} is given as:

$$R_{PI} = \{<, s, d, f\} \text{ or } R_{PI} = \{\text{less than, start, during, finish}\}$$

The symbol '?' can be used to represent an unknown relationship

Functions: The following two functions are used to represent quantitative information associated with intervals.

The *Interval length function* assigns a non-zero positive real number to a system interval.

$$\text{Length } X = d, \text{ where } X = [sX, eX],\ d \in \Re^+$$

A recent extension to PIL enables specification of lower and upper bounds on an interval length. The two bounds can also be represented with the help of the *at least* and *at most* temporal relations.

Length X ≥ d, where X = [sX, eX], d $\in \Re^+$ (d is a lower bound on length)

Length X ≤ d, where X = [sX, eX], d $\in \Re^+$ (d is an upper bound on length)

The *stamp function* assigns a non-negative real number to a point. Recent extensions to PIL enable specification of lower and upper bounds on a point stamp.

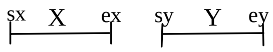
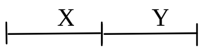
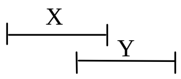
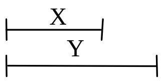
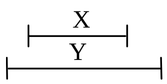
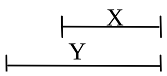
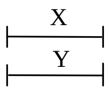
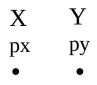
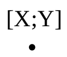
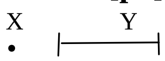
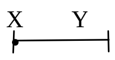
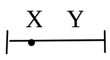
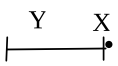
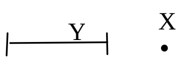
Stamp p = t, where t $\in \Re^+ \cup \{0\}$

Stamp p ≤ t, t $\in \Re^+ \cup \{0\}$

Stamp p ≥ t, t $\in \Re^+ \cup \{0\}$

In Table 2.1 we show the syntactic and semantic structure of PIL expressions. Note that each relationship between intervals or an interval and a point can be constructed with the help of inequalities between their start and end points.

Table 2.1 PIL Expressions and Their Semantics

CASE I— X and Y both intervals with non-zero lengths: X = [sx, ex], Y = [sy, ey] with sx < ex and sy < ey				
1. X < Y	ex < sy			
2. X m Y	ex = sy			
3. X o Y	sx < sy,	sy < ex,	ex < ey	
4. X s Y	sx = sy,	ex < ey		
5. X d Y	sx > sy,	ex < ey		
6. X f Y	sx > sy,	ey = ex		
7. X = Y	sx = sy,	ex = ey		
CASE II—X and Y both points: X = [px] and Y = [py]				
1. X < Y	px < py			
2. X = Y	px = py			
CASE III— X is a point and Y is an interval: X = [px] and Y = [sy, ey]				
1. X < Y	px < sy			
2. X s Y	px = sy			
3. X d Y	sy < px < ey			
4. X f Y	px = ey			
5. Y < X	ey < px			

A graph construct called Point Graphs (PG) is used as an underlying structure to represent statements in PIL. In a PG, a node represents a point (or a *composite* point) and an edge between two points represents one of the two temporal relations, less than and less-than-or-equal, between the two. Two or more points pi, pj, …, pn are represented as a composite point [pi; pj; …; pn], or a single node in a PG, if all are mapped to a single point on the timeline. The statements in PIL can be converted to an equivalent PG representation with the help of the corresponding analytic inequalities shown in Table 2.1. In addition, the quantitative temporal information, modeled using the length and the stamp function, is represented as node and arc inscriptions on the PG. All the verification, revision and inference algorithms work by manipulating this Point Graph representation of the set of PIL statements. We show in Figure 2.1 how a set of PIL statements can be converted into a Point Graph.

Point p1, p2, p3
p1 < p2
p2 ≤ p3
Stamp p1 = 4500
Length [p1, p2] = 100

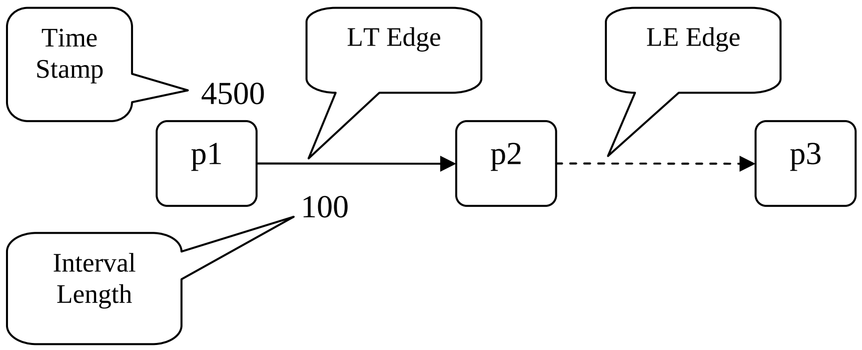


Figure 2.1: Point graph representation of a set of PIL statements

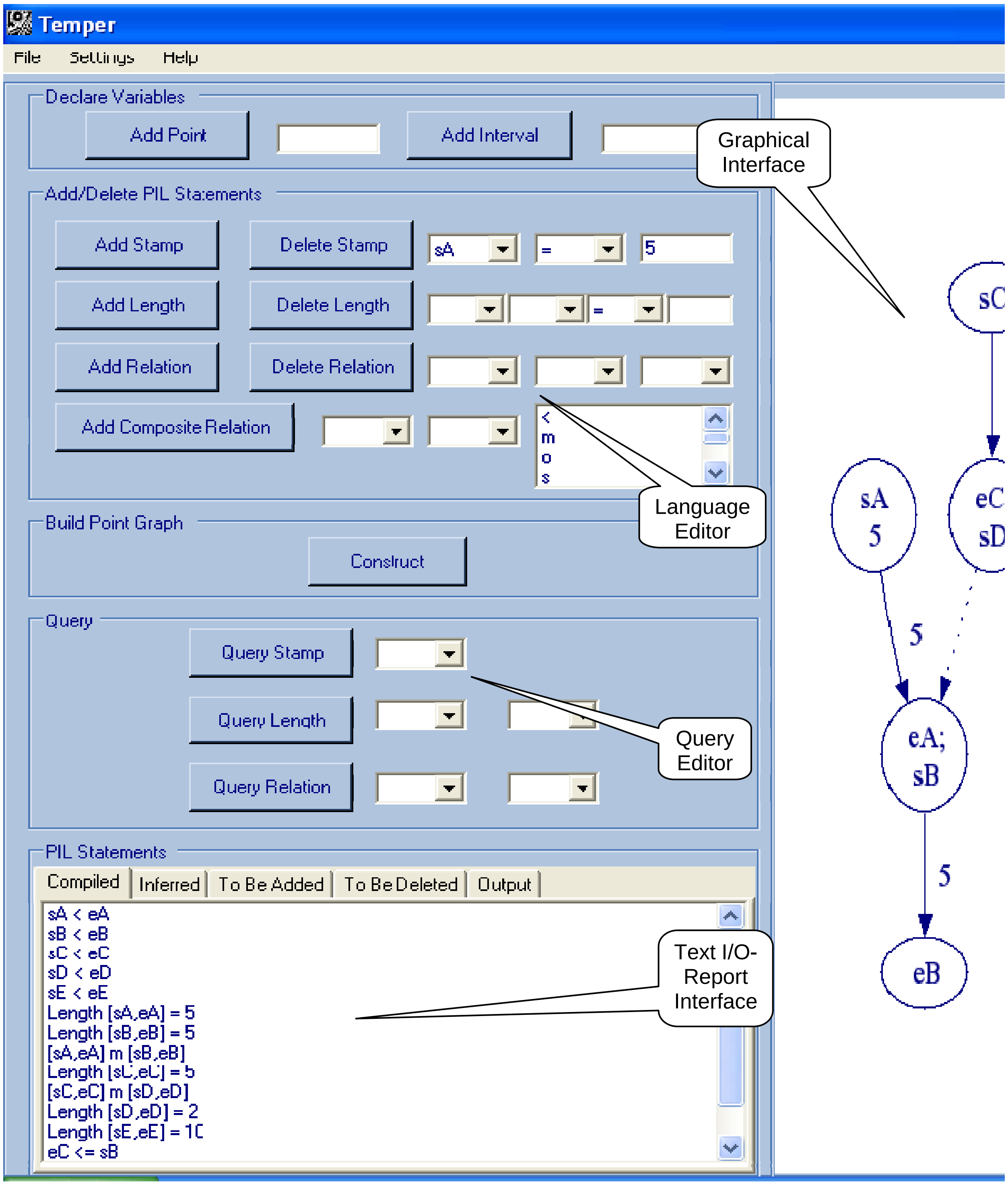
In this paper we explore an application of PIL and its associated algorithms to criminal forensic analysis. A detailed description of Point-Interval Logic and Point Graphs can be found in Zaidi (1999); Zaidi and Levis (2001); and Zaidi and Wagenhals (2006).

3 Temper: The Software Tool

A software tool called Temper (Temporal Programmer) implements the inference mechanism of Point-Interval Logic along with its verification and revision mechanisms. Temper provides a language editor to input PIL statements and a query editor to run various queries on the constructed Point Graph. It has a graphical interface to display the Point Graph and also a text I/O interface to display information and results of the analyses. Figure 3.1 shows the user interface of Temper. In the PGs shown, each point is represented as a node, and each interval is represented by two nodes connected by a less than (<) or LT arc. Each LT arc is represented by a solid arc and the length, if available, appears adjacent to the arc. Each less-than-or-equal (≤) or LE edge is represented by a dotted arc. The stamp on each point appears inside the node representing the point. A special type of node, called virtual node, is used to represent *at least*, *at most*, *no later than*, *or no earlier than* temporal relations. Figure 3.2 shows how the various elements of a Point Graph are displayed in Temper.

4 Illustrative Example: London Bombing Investigation

On July 7, 2005, there were four explosions in London at Travistock Square, Edgware Road, Algate, and Russell Square. Three of these explosions, Edgware Road, Algate, and Russell Square, took place in trains that departed from King-Cross station. Images from close-circuit cameras installed at London's various railway stations were an important source of information for investigators. There were hours of images available from these cameras and the task of investigators was to analyze these images to identify possible suspects. The large number of such images, although desirable, can make an investigation that requires searching through them in a timely manner very time consuming.



sA < eA
sB < eB
sC < eC
sD < eD
sE < eE
Length [sA,eA] = 5
Length [sB,eB] = 5
[sA,eA] m [sB,eB]
Length [sC,eC] = 5
[sC,eC] m [sD,eD]
Length [sD,eD] = 2
Length [sE,eE] = 10
eC <= sB

Figure 3.1. Temper User Interface

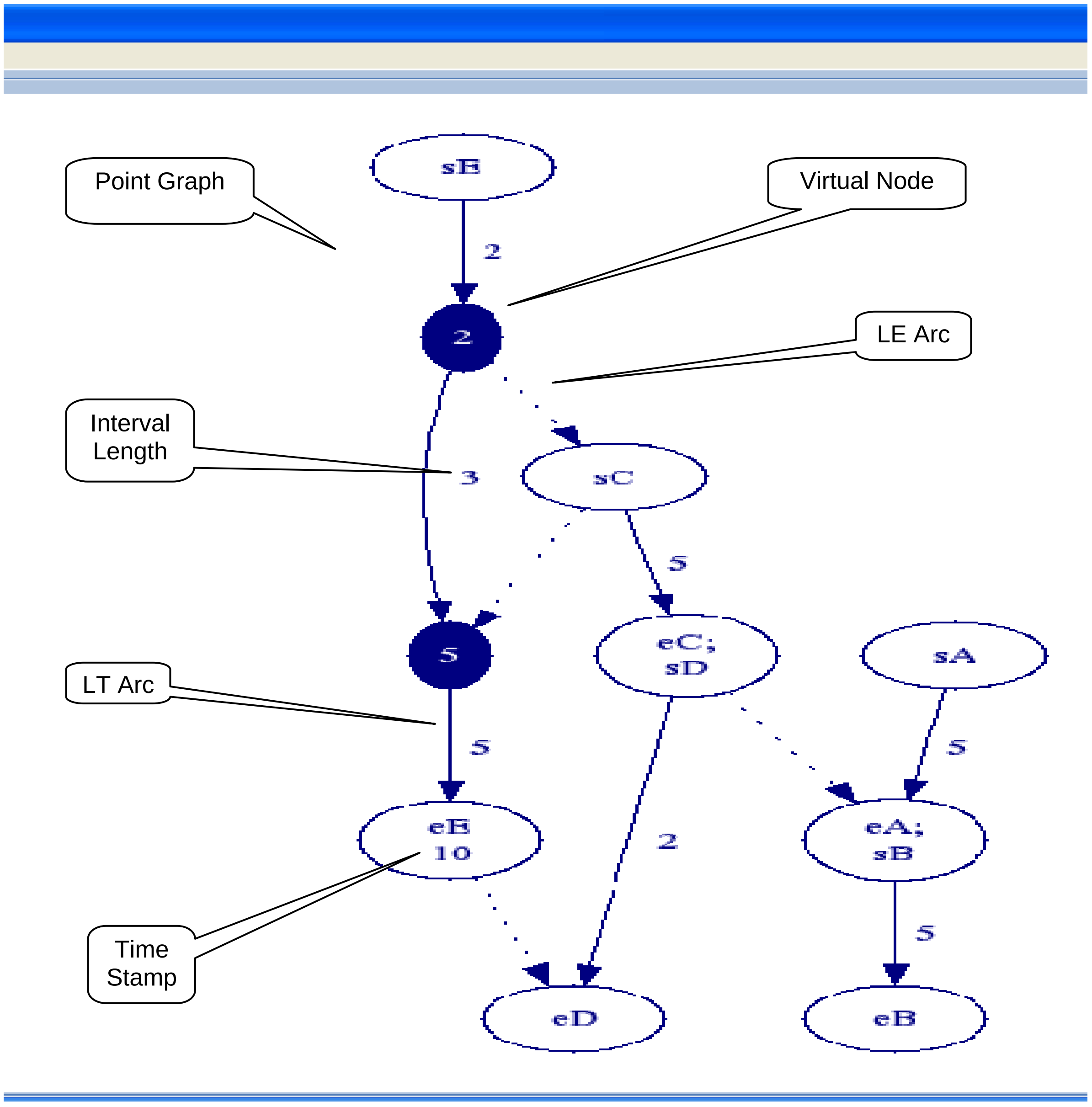


Figure 3.2. Point Graph Visualization in Temper

In this section, we will demonstrate how Temper can be used to restrict the size of a potential interval for which to analyze images (by making sense of the available temporal information.) and thus speeding up the investigation. Since Temper has the ability to handle both qualitative and quantitative constraints, both types of information regarding the incident and/or the surrounding events can be input to it. Temper also offers the additional advantage of the verification mechanism that can be invoked to check the consistency of the available temporal information. This can be very useful when temporal information may originate from multiple (and possibly unreliable) sources. In this example, we will demonstrate the capabilities of Temper by modeling a set of temporal information items related to the incident and by trying to identify the exact time or the shortest possible interval during which one of the ill-fated trains left from King-Cross station for Edgware.

The journey of the three trains from King-Cross station can be represented as PIL intervals. The journey of these trains ended in explosions. We also know the lower bounds on the travel times of these trains after their departures from King-Cross station, based on the distances of the sites of the explosions from King-Cross station. The train from King-Cross to Edgware must have traveled for at least 5 time units. Similarly trains to Algate and Russell Square must have traveled for 4 and 5 time units, respectively. The time units are defined by a mapping from actual clock times to an equivalent representation on a real number line. Table 4.1 shows how this information can be represented as PIL statements. These PIL statements are input to Temper using its language editor. Figure 4.1 shows the corresponding Point Graph in Temper.

Table 4.1: PIL Statements for London Bombing Scenario

Temporal Information	**PIL Statements**
Train traveling from King-Cross to Edgware	*interval* Train_KingX_Edgware
Train traveling from King-Cross to Algate	*interval* Train_KingX_Algate
Train traveling from King-Cross to Russell Square	*interval* Train_KingX_Russell_Sq
Explosion at Edgware	*point* Explosion_Edgware
Explosion at Algate	*point* Explosion_Algate
Explosion at Russell Square	*point* Explosion_Russell_Sq
Explosion at Edgware ended the journey of train from King-Cross to Edgware	Explosion_Edgware *f* Train_KingX_Edgware
Explosion at Algate ended the journey of train from King-Cross to Algate	Explosion_Algate *f* Train_KingX_Algate
Explosion at Edgware ended the journey of train from King-Cross to Russell Square	Explosion_Russell_Sq *f* Train_KingX_Russell_Sq
Train from King-Cross to Edgware traveled at least for 5 time units	*Length* [Train_KingX_Edgware] $\geq$ 5
Train from King-Cross to Algate traveled at least for 4 time units	*Length* [Train_KingX_Algate] $\geq$ 4
Train from King-Cross to Russell Square traveled at least for 5 time units	*Length* [Train_KingX_Russell_Sq] $\geq$ 5

Once the temporal information has been input, Temper can be used to draw inferences about the point of interest, i.e., the instant when one of the trains left King-Cross station for Edgware. We run a query, using the query editor of Temper, for the time stamp of the point "sTrain_KingX_Edgware" which represent the departure of the train from King-Cross station to Edgware. Figure 4.2 shows the query in Temper and Figure 4.3 shows the result of the query.

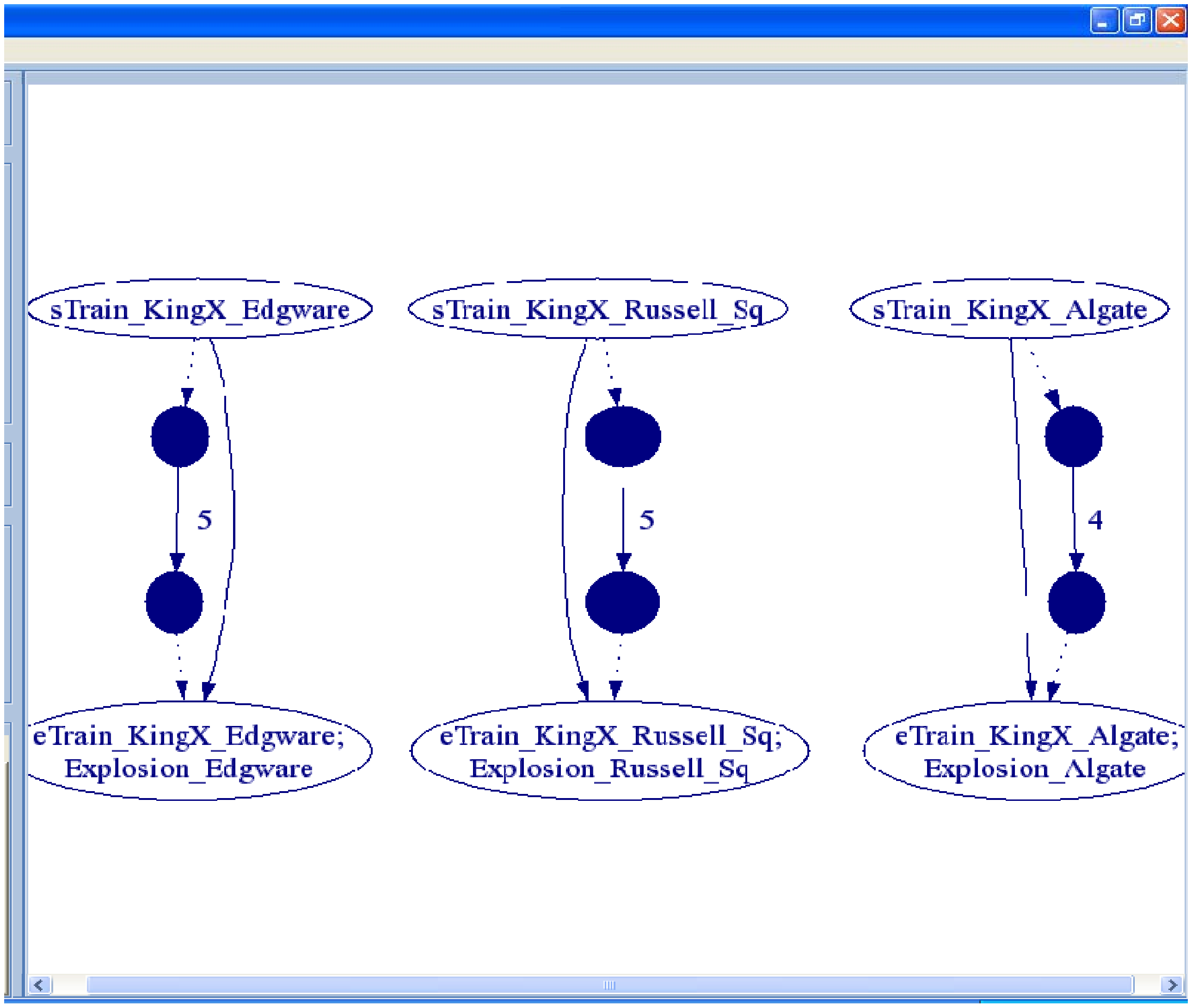

sTrain_KingX_Edgware
sTrain_KingX_Russell_Sq
sTrain_KingX_Algate
5
5
4
eTrain_KingX_Edgware; Explosion_Edgware
eTrain_KingX_Russell_Sq; Explosion_Russell_Sq
eTrain_KingX_Algate; Explosion_Algate

Figure 4.1. Point Graph for London Bombing Scenario

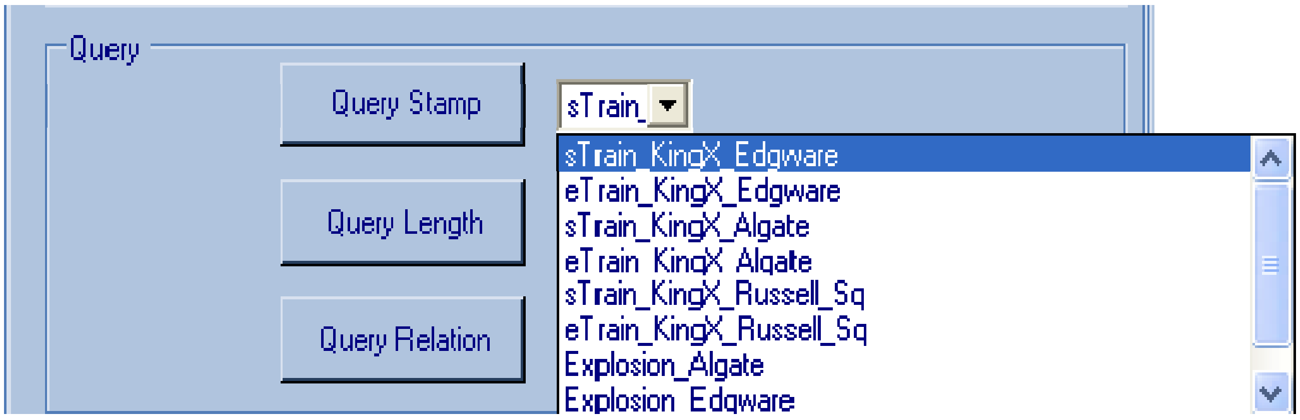


Figure 4.2. Running a Query in Temper

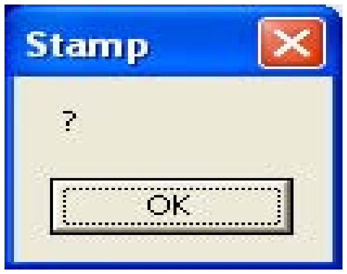


Figure 4.3. Result of the Query

For the illustrated case, Temper cannot infer anything about the stamp of the event based on the information provided so far (Figure 4.3). Suppose, further investigation reveals that the explosion near Edgware took place between time units 840 and 845 (the explosion is considered to be an instantaneous event so the range 840 to 845 does not represent duration but the uncertainty in determining the actual occurrence time). Similarly the explosions near Algate and Russell Square occurred between 845 and 850, and between 840 and 850 respectively. These times are not actual clock times, rather their equivalent representation obtained by mapping the clock times on a real number line. Table 4.2 shows how this information can be represented as PIL statements. These PIL statements are added to the initial temporal model to get the Point Graph of Figure 4.4.

Once again, the query for the time stamp of the point "sTrain_KingX_Edgware" is executed. As can be seen in Figure 4.5, Temper was able to determine an upper bound for the stamp of the event, i.e., the train from King-Cross to Edgware must have left no later than 847.

Table 4.2: Additional PIL Statements for London Bombing Scenario

Temporal Information	PIL Statements
Explosion at Edgware happened no earlier than 840	*Stamp* [Explosion_Edgware] ≥ 840
Explosion at Edgware happened no later than 852	*Stamp* [Explosion_ Edgware] ≤ 852
Explosion at Algate happened no earlier than 845	*Stamp* [Explosion_Algate] ≥ 845
Explosion at Algate happened no later than 850	*Stamp* [Explosion_Algate] ≤ 850
Explosion at Russell Sq. happened no earlier than 840	*Stamp* [Explosion_Russell_Sq] ≥ 840
Explosion at Russell Sq. happened no later than 850	*Stamp* [Explosion_Russell_Sq] ≤ 850

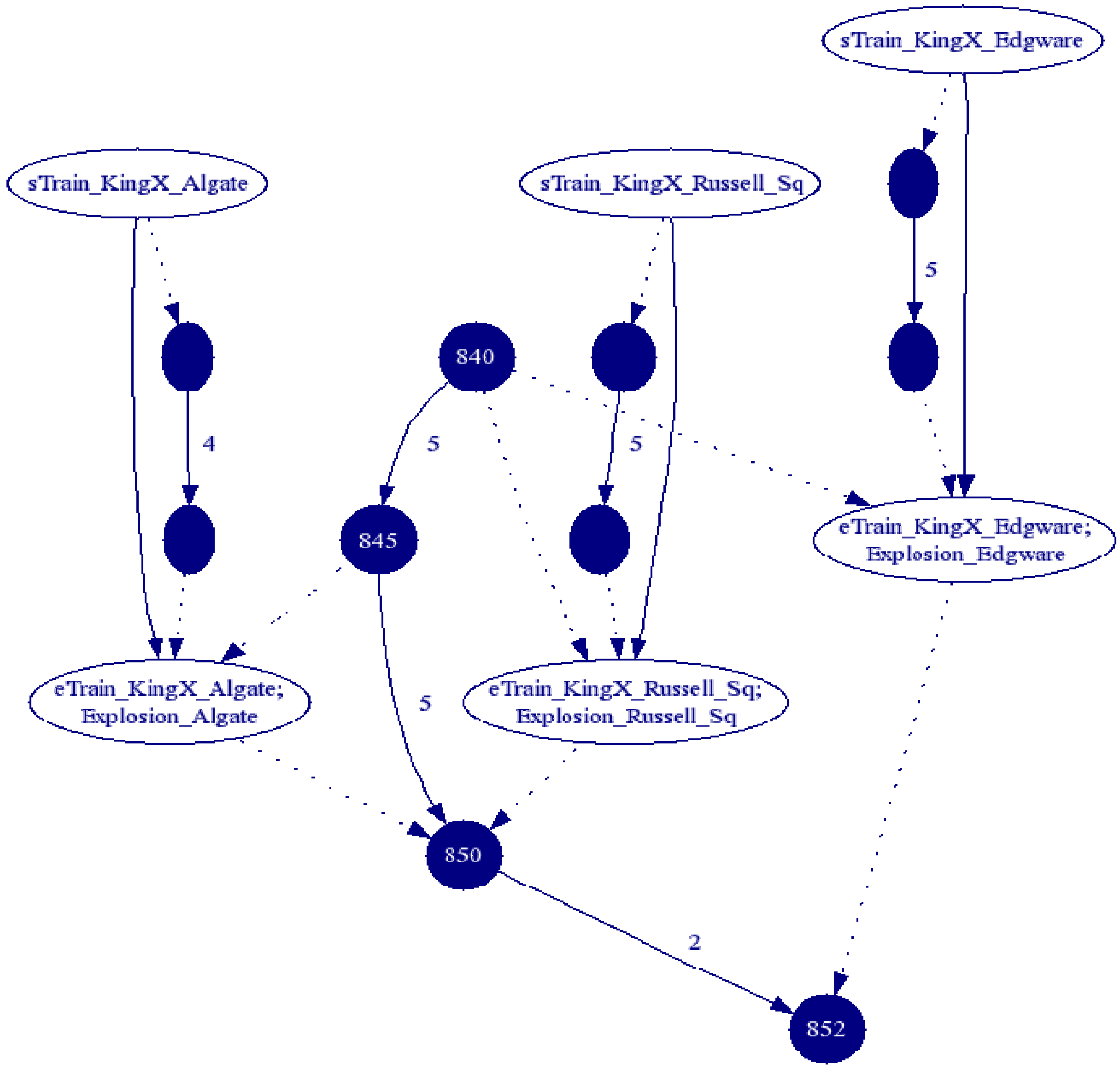
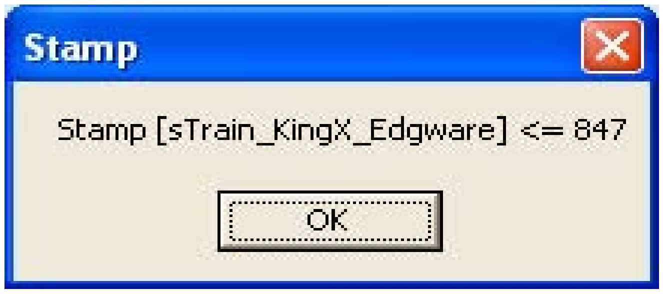


Figure 4.4. Revised Point Graph for London Bombing Scenario



Stamp [sTrain_KingX_Edgware] <= 847

Figure 4.5. Result of the Query

As indicated earlier, the verification mechanism of Temper can detect the inconsistencies in the available temporal information. The ability to detect inconsistency can be very useful, when the information from different sources is combined into a single model of the situation. Suppose, we input to Temper the information that the train from King-Cross to Edgware left at time instant 848 (represented by the PIL statement: Stamp[sTrain_KingX_Edgware] = 848). Clearly, this statement is in conflict with the previously added PIL statements. Temper detects this inconsistency (manifested in the form of inconsistent paths in the Point Graph, shown with an extra boundary around each node in the path), and identifies the portion of the Point Graph that contains the contradiction, as shown in Figure 4.6. Note the two inconsistent paths (from node with stamp 840 to node with stamp 852): one path with length exactly equal to 12 units and the other path having a length of at least 13 units. We fix this inconsistency by deleting the statement "Stamp [sTrain_KingX_Edgware] = 848".

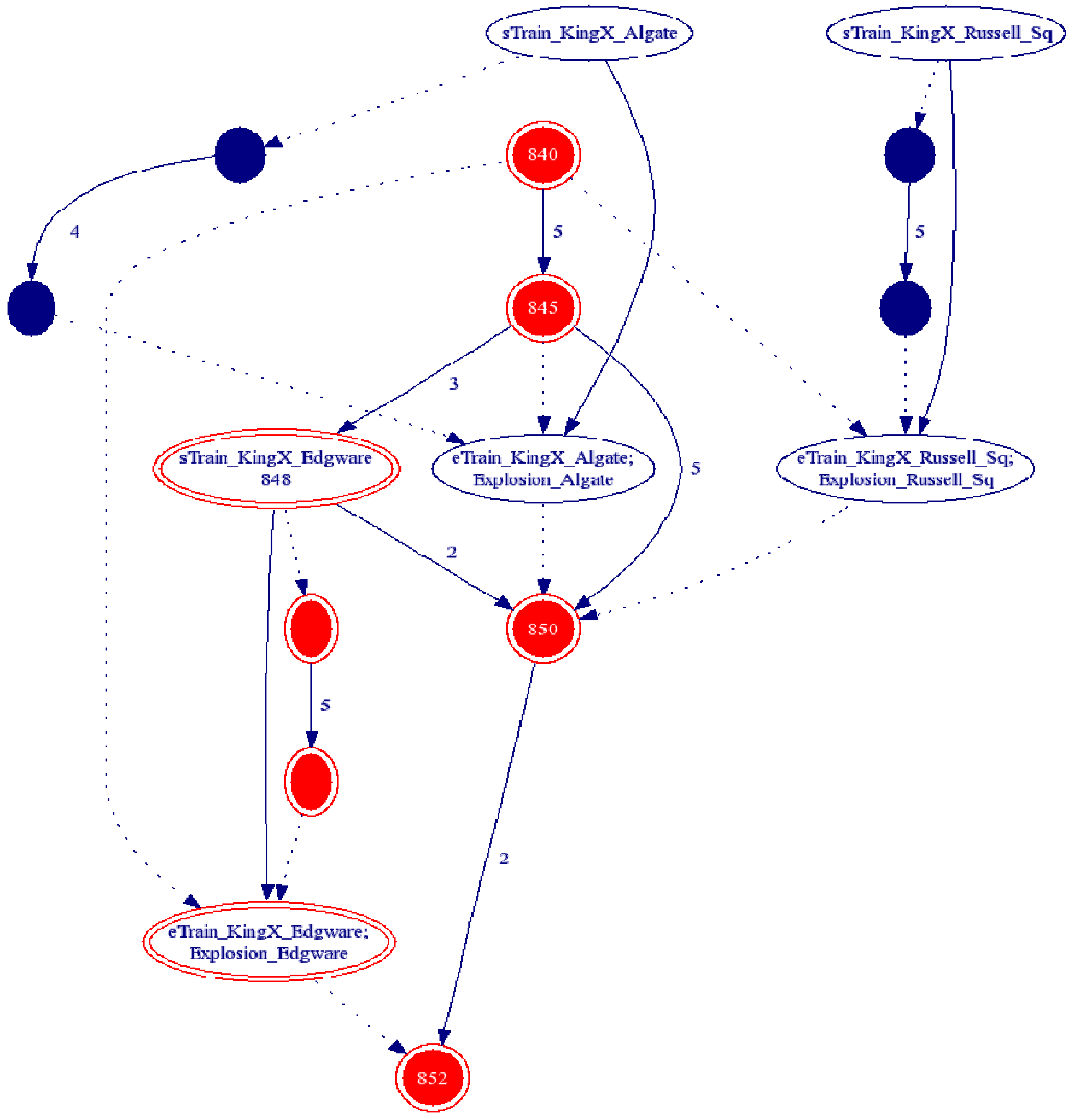


Figure 4.7. Inconsistency in the Point Graph

Suppose that the investigators have also identified four suspects who were spotted entering the Luton railway station at time instant 720. The investigators believe that these suspects took a train from Luton to King-Cross station, and at King-Cross station they boarded the trains in which the explosions took place. The next train from Luton to King-Cross departed at 748 and reached King-Cross at time instant 842. Obviously, if these suspects were in fact the bombers, the train from Luton should have reached King-Cross before any train in which there was an explosion left King-Cross station. This information can be represented by PIL statements as given in Table 4.3 and the resulting PG is shown in Figure 4.8. Note that Table 4.3 contains both qualitative and quantitative PIL statements.

The query for the time stamp of the point "sTrain_KingX_Edgware" is executed one more time. Figure 4.9 presents the result of the query; Temper was able to determine both an upper bound and a lower bound for the stamp of the event, i.e., the train must have left King-Cross station after time instant 842 and no later than 847. Note that the lower bound is strict. Thus by applying the Point-Interval Logic to the analysis of available temporal information, we have identified the bounds of the interval that we were interested in. The images need be analyzed for this interval only; this improves the timeliness of the labor intensive image analysis process.

Table 4.3: Additional PIL Statements for London Bombing Scenario

Temporal Information	**PIL Statements**
Train traveling from Luton to King-Cross station	*interval* Train_Luton_KingX
Suspected spotted entering the Luton station	*point* Suspects_Spotted_at_Luton
Suspected spotted at Luton at time instant 720	*Stamp* [Suspects_Spotted_at_Luton] = 720
Train from Luton to King-Cross left at 748	*Stamp* [sTrain_Luton_KingX] = 748
Train from Luton to King-Cross arrived at 842	*Stamp* [eTrain_Luton_KingX] = 842
Train to Edgware left after the train from Luton	eTrain_Luton_KingX < Train_KingX_Edgware
Train to Algate left after the train from Luton	eTrain_Luton_KingX < Train_KingX_Algate
Train to Russell Sq. left after the train from Luton	eTrain_Luton_KingX < Train_KingX_Russell_Sq

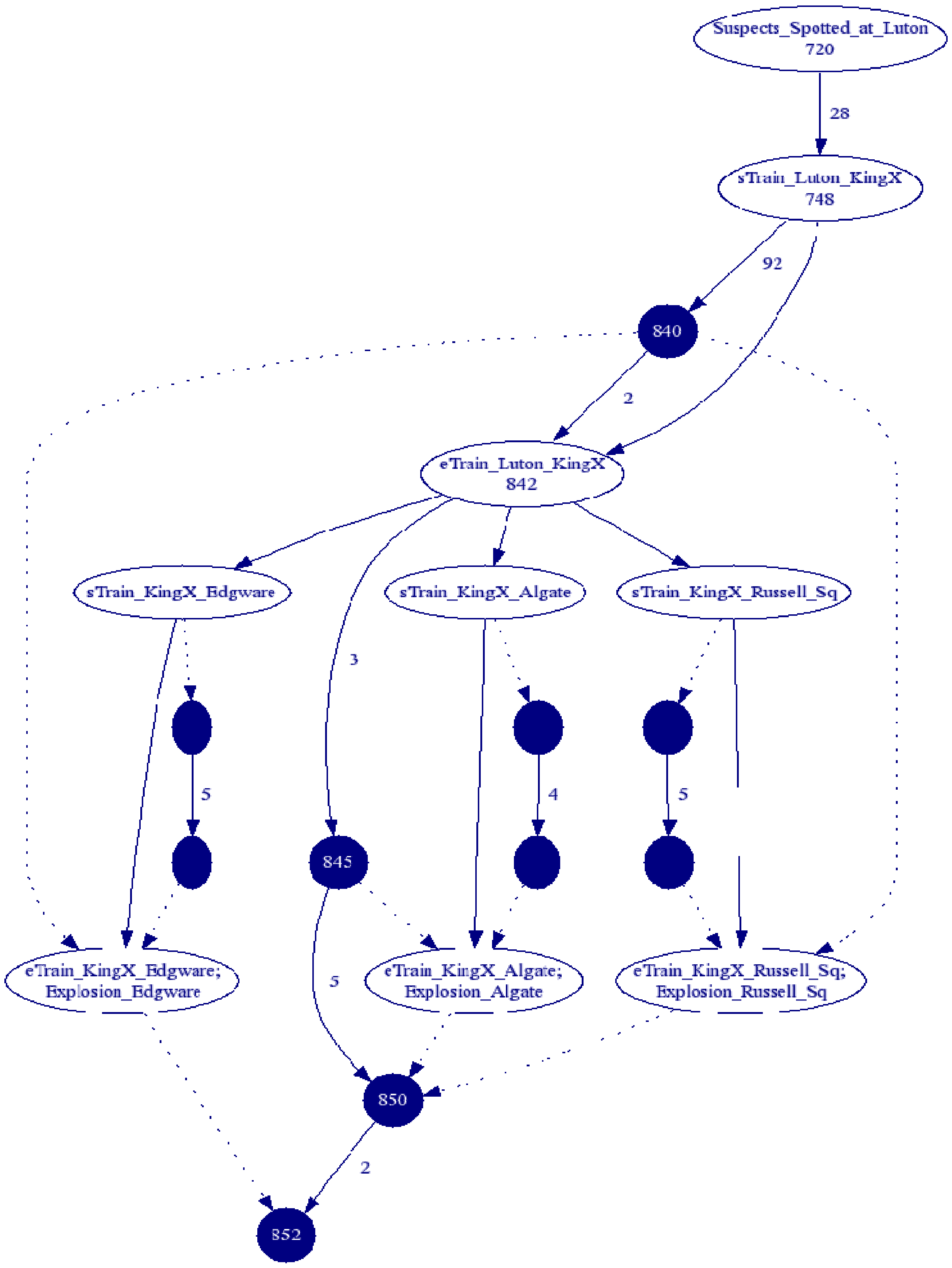
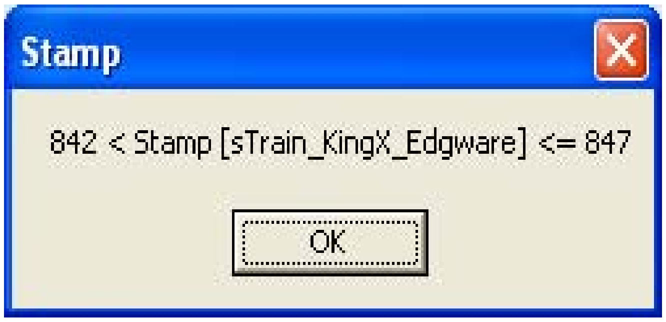


Figure 4.8. Revised Point Graph for London Bombing Scenario



```
842 < Stamp [sTrain_KingX_Edgware] <= 847
```

Figure 4.9. Result of the Query

5 Conclusion

This paper presented an illustration of how the Point-Interval Logic (PIL) can be used to create temporal models of situations arising in forensics and help investigators answer interesting questions. The approach was demonstrated using Temper, which is a software implementation PIL, and the London bombing incident as the scenario. We showed how Temper could be used to answer meaningful questions during an investigation, in this case identifying a small interval for which the images from close-circuit cameras should be analyzed. The reader may argue that the problem could have been solved manually as well; that is true in the case of a small example like this one but in general situations the set of temporal statement may be too large for a human to handle. This calls for a computer-aided approach to such an analysis. In addition, Temper can combine temporal information from multiple sources, detect inconsistencies and identify the specific source(s) with inconsistent information. Temper can, therefore, be used to compare witness accounts of several individuals on the same incident for overlaps and inconsistencies—another useful application for forensics.

References

Allen, J. F. 1983. Maintaining Knowledge About Temporal Intervals. *Communications of ACM* 26:832-843.

Hamblin, C. L. (1972). Instants and Intervals. In F. Haber J. Fraser & G. Muller (eds.), *The Study of Time*. New York: Springer Verlag, 324–328.

Ladkin, P. B., and Maddux, R. 1988. On binary constraint networks, Technical Report, KES.U.88.8, Kestrel Institute, Palo Alto, Calif.

Peterson, J. L. (1981). *Petri Net Theory and the Modeling of Systems*. Prentice-Hall.

Rauf, I., and Zaidi, A. K. (2002). A Temporal Programmer for Revising Temporal Models of Discrete-Event Systems, in: *Proc. of 2002 IEEE International Conference on Systems, Man, and Cybernetics*, Hemmamat, Tunisia.

Reisig, W. (1991). Petri Nets and Algebraic Specifications. Theoretical Computer Science 80(1): 1-34 .

Zaidi, A. K. 1999. On Temporal Logic Programming Using Petri Nets. *IEEE Transactions on Systems, Man and Cybernetics*, Part A, 29(3): 245-254.

Zaidi, A. K., and Levis, A. H. 2001. TEMPER: A Temporal Programmer for Time-sensitive Control of Discrete-event Systems. *IEEE Transaction on Systems, Man, and Cybernetic,* 31(6):485-496.

Zaidi, A. K., and Wagenhals, L. W. 2006. Planning Temporal Events Using Point-Interval Logic. Special Issue of *Mathematical and Computer Modeling*. Forthcoming.